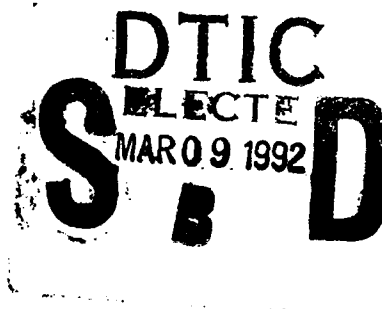


2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A247 036



THESIS

EFFECTS OF NON-NORMAL OUTLIER-PRONE ERROR
DISTRIBUTION ON KALMAN FILTER TRACK

by

Jose Batista de Souza Neto

Thesis Advisor:

Donald P. Gaver

Approved for public release; distribution is unlimited

92 3

92-05868



REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School		6b. OFFICE SYMBOL (If applicable) 55		7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000				7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION		8b. OFFICE SYMBOL (If applicable)		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code)		10. SOURCE OF FUNDING NUMBERS			
		Program Element No		Project No	Task No
					Work Unit Accession Number
11. TITLE (Include Security Classification) EFFECTS OF NON-NORMAL OUTLIER-PRONE ERROR DISTRIBUTION ON KALMAN FILTER TRACK					
12. PERSONAL AUTHOR(S) Souza Neto, Jose, B.					
13a. TYPE OF REPORT Master's Thesis		13b. TIME COVERED From To		14. DATE OF REPORT (year, month, day) 1991, September	
				15. PAGE COUNT 69	
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17. COSATI CODES			18. SUBJECT TERMS (continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUBGROUP	Kalman Filter		
19. ABSTRACT (continue on reverse if necessary and identify by block number) This report is about the effects of non-normality on the efficiency of the Kalman filter, particularly when the distribution of measurement errors is still symmetric but the tails are extended, which means that the observations are outlier-prone.					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS REPORT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Donald P. Gaver			22b. TELEPHONE (Include Area code) (408)6462605		22c. OFFICE SYMBOL OR/GV

Effects of Non-Normal Outlier-Prone Error Distribution
on
Kalman Filter Track

by

Jose Batista de Souza Neto
Lieutenant, Brazilian Navy
B.S., Brazilian Naval Academy, 1979

Submitted in partial fulfillment
of the requirements for the degree of

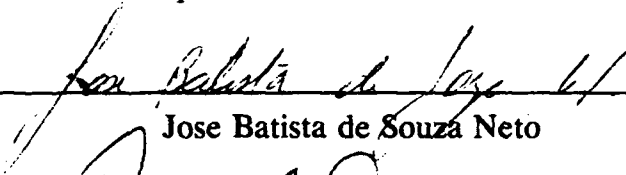
MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

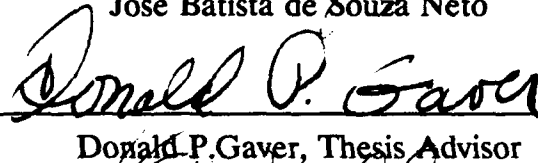
NAVAL POSTGRADUATE SCHOOL

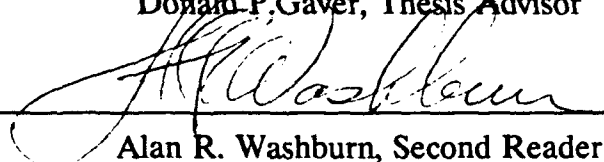
September 1991

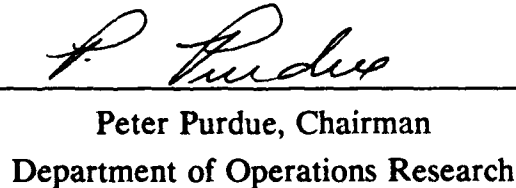
Author:


Jose Batista de Souza Neto

Approved by:

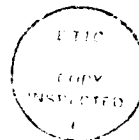

Donald P. Gaver, Thesis Advisor


Alan R. Washburn, Second Reader


Peter Purdue, Chairman
Department of Operations Research

ABSTRACT

This report is about the effects of non-normality on the efficiency of the Kalman filter, particularly when the distribution of measurement errors is still symmetric but the tails are extended, which means that the observations are outlier-prone.



Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION	1
II. BACKGROUND	3
A. DIGITAL FILTERS	3
1. Recursive Filters	4
B. MOTION MODELS	15
1. The Ornstein-Uhlenbeck Process	16
III. MODEL DESCRIPTION	19
A. DESCRIPTION OF THE ONE SOURCE (SINGLE TARGET) MODELS	19
1. The Range, Bearing, Radial and Angular Velocity Model (Filter Model 1)	19
2. The (X, Y, V_x , V_y) Simple Model (Filter Model 2)	22
3. The (X, Y, V_x , V_y) Model Using I.O.U. Process (Filter Model 3). .	24
B. DESCRIPTION OF THE 2 SOURCES (SINGLE TARGET) MODEL	26
IV - MODEL EVALUATION	28
A. TARGET MOTION GENERATOR	28

1. Target Measurement Error	28
2. Target Motion Pattern.	33
B. EVALUATION.	35
1. Experiment 1: Inbound Target	36
2. Experiment 2: Crossing Target	39
3. Experiment 3: Random Tour Target	41
V - CONCLUSIONS AND RECOMMENDATIONS.	47
A. SUMMARY	48
APPENDIX A - FORTRAN PROGRAM FOR FILTER MODEL 3	49
APPENDIX B - FORTRAN PROGRAM FOR THE MOTION GENERATOR	54
LIST OF REFERENCES	61
INITIAL DISTRIBUTION LIST	62

I. INTRODUCTION

Most contemporary automatic target trackers use a Kalman filter in their kernels. One of the main assumptions behind the Kalman filter algorithm is that the errors in the observations of the target are normally distributed. This report is about the effects of non-normality on the efficiency of the Kalman filter, particularly when the distribution of measurement errors is still symmetric but the tails are extended, which means that the observations are outlier-prone.

Chapter II explains the bases of digital filters (in particular the Kalman filter) and the Integrated Ornstein-Uhlenbeck motion model, which are the tools for this study. Chapter III describes the three models that will be used in the experiments. A fourth model is presented to demonstrate the data fusion of two sources of information. Chapter IV describes the experiments and analyses the results and Chapter V summarize the conclusions.

The "real world" situation that is inspiring our model can be described as follows: a station equipped with an Electronic Support Measure (ESM) equipment is trying to track a target by measuring its bearing and signal intensity. Those two measures have different inherent error, intrinsically characteristic of the equipment and of the physical conditions. The signal intensity provides the "range measurement" based on emission characteristics (such as transmission power, frequency, etc.) and propagation conditions. The conversion of signal intensity into range is certainly the largest source of error for the target state estimate. The bearing is measured directly in the

equipment, and we expect it to be more accurate than range measurement; the exact precision is an equipment attribute (Figure 1).

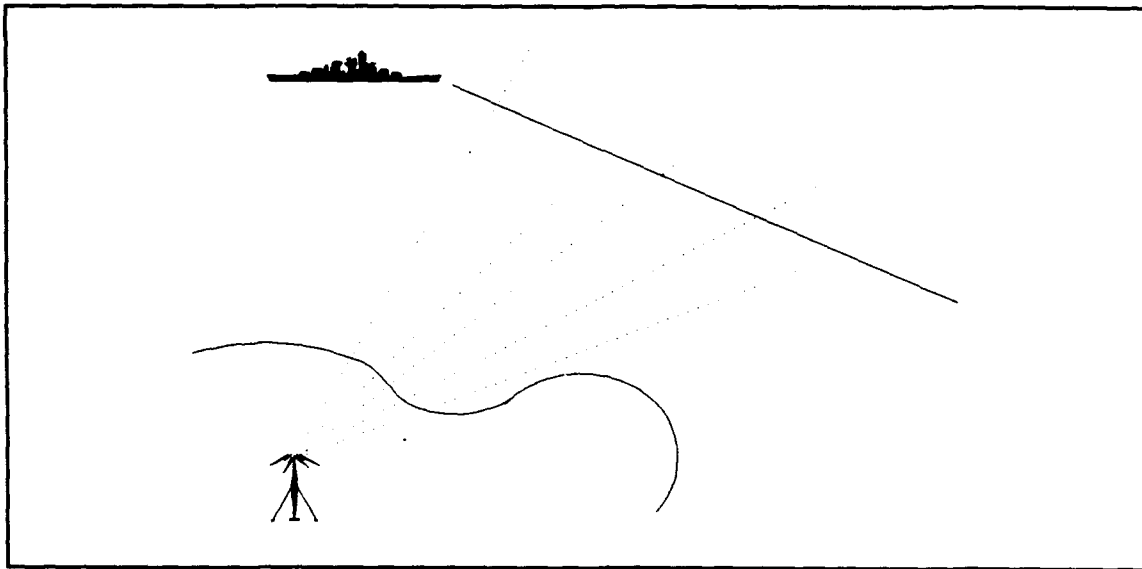


Figure 1 - Hypothetical situation

In the situation described, large relative range errors and small bearing errors should be expected.

II. BACKGROUND

This chapter was inserted in the thesis to help the reader who is not familiar with the Kalman filter and the Integrated Ornstein-Uhlenbeck model. For the reader who is already familiar with these subjects, this section is optional. The following is a synopsis of various documents [Ref. 1 through 4].

A. DIGITAL FILTERS¹

A sequence of bearings and ranges is produced when periodic measures of a target are made by a Radar, Sonar or other detection device. Those observations carry in addition to their inaccuracy an associated uncertainty which is usually represented by additive noise [Ref. 5].

The algorithms used to process those measurements are called *Digital Filters*. If we call the inputs $IN(t)$ and the outputs $OUT(t)$, Equation 1 represents a digital filter

$$OUT(t) = \sum_{i=0}^{\infty} C_i IN(t-i) + \sum_{i=1}^{\infty} D_i OUT(t-i) \quad (2.1)$$

Where the C_i 's and D_i 's are constants. The formula says that the filter is a linear combination of measures and previous values obtained in the filter. If all coefficients D_i are null, the filter is called *non-recursive*. In this thesis we are interested in a recursive filter, the Kalman Filter, which uses the last measurement and a combination of all the previous ones.

¹ This section is an adapted translation of Reference 1.

The name "filter" comes from the analogy with the idea of a filter in electronics, where for example, when we want to eliminate high frequency signals we use a "Low Pass" filter, and only slow changes in the signal are allowed to go through (Figure 2). Low pass characteristics are obtained on the filter in Equation 2.1 when $D_i \gg C_i$.

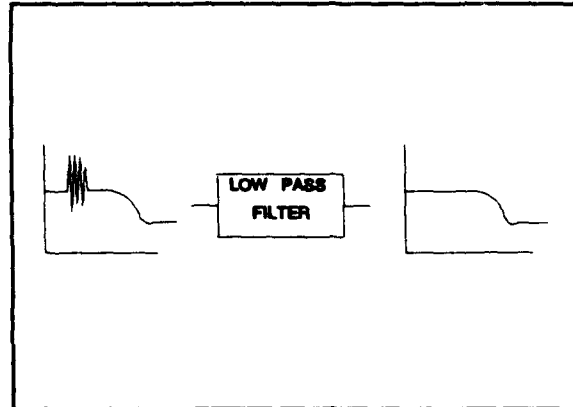


Figure 2 - Electronic filter

Analogously, our filter will compare the measure with the previous state of the system, allowing the last measure to have more or less influence on the system state change, according to the gain established.

1. Recursive Filters

As mentioned previously, a recursive filter uses the last measurement and a combination of the previous measures that is in the system state. As an example of a recursive algorithm, look at the equation

$$OUT(t) = \left(1 - \frac{1}{K}\right) IN(t) + \frac{1}{K} OUT(t-1) \quad (2.2)$$

Considering the input at $t=0$ as $IN(0)$ and $OUT(t)=0$ for $t<0$, then the sequence of values will be generated at each unitary time step:

$$OUT(0) = IN(0)$$

$$OUT(1) = (1 - \frac{1}{K}) IN(1) + \frac{1}{K} IN(0)$$

$$OUT(2) = (1 - \frac{1}{K}) IN(2) + \frac{1}{K} OUT(1) \quad (2.3)$$

$$= (1 - \frac{1}{K}) IN(2) + (\frac{1}{K} - \frac{1}{K^2}) IN(1) + \frac{1}{K^2} IN(0)$$

Note that the value $IN(0)$ is never forgotten and can have more or less influence in the present state depending on the value of K . Note also that the value of $OUT(t-1)$ economically condenses all the previous information up to time $t-1$.

a. Filters α_β and α_β_γ

Consider a stationary target (order zero system) for which measures of one coordinate are being taken. One filter that might be used to represent the system is

$$\begin{aligned} OUT(t) &= \alpha IN(t) + (1-\alpha) OUT(t-1) \\ &= OUT(t-1) + \alpha(IN(t) - OUT(t-1)) \end{aligned} \quad (2.4)$$

In this case the actual state (target position) is obtained by combining the previous state with the difference between the measurement and the previous System state, multiplied by the gain α . If the gain $\alpha = 1$ total confidence is placed in the measure and the previous state is forgotten. Conversely $\alpha = 0$ will not use the measure, keeping the previous state. This is called the α filter.

Consider now a target with constant velocity (first order system)(Figure 3). The filter now has to predict the position at the time t of the measure before comparing it with the estimate. For that system, calling X the target position and V_x

the target velocity, using the same idea as for the previous filter the equations are:

$$\text{PREDICTION} \quad X_p(t) = X_e(t-\Delta t) + V_x \Delta t \quad (2.5)$$

$$\text{ESTIMATION} \quad X_e(t) = X_p(t) + \alpha(X_m(t) - X_p(t)) \quad (2.6)$$

where X_p = predicted position

X_m = measured position

X_e = estimated position

α = filter gain

That means the estimated position is placed somewhere between the predicted position and the measured position, using a linear combination.

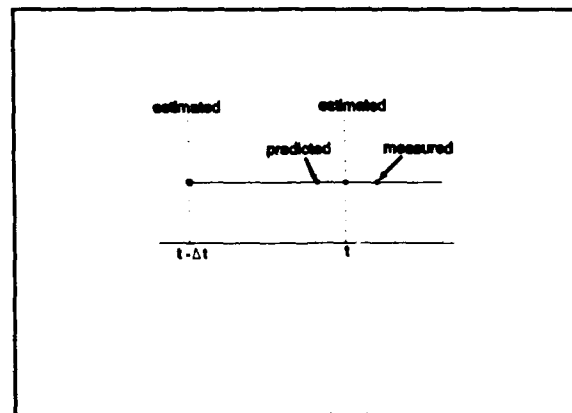


Figure 3 - Prediction Step.

It is possible now, using the previous estimated position and the measured position, to calculate the measured velocity. Then a linear combination of the measured and predicted velocities is formed using a new weighing parameter, the gain β , to achieve a better estimate of velocity. Remember that the predicted velocity at time t is equal to the estimated velocity at time $t-\Delta t$ by assumption. The equation for the estimated

velocity is:

$$V_e(t) = V_p(t) + \beta(V_m(t) - V_p(t)) \quad (2.7)$$

or, in a continuous time formulation

$$\frac{d}{dt}X_e(t) = \frac{d}{dt}X_e(t-\Delta t) + \frac{\beta}{\Delta t}(X_m(t) - X_p(t)) \quad (2.8)$$

Note that when the measure is greater than prediction, the correction in the estimated velocity is positive (as it should be). The parameter β is the filter velocity gain and the filter is called α_β . In the same way a filter for a system with constant acceleration (second order system) might be derived. A new gain factor will appear and the filter will be called α_β_γ [Ref. 1].

Looking at the filter α_β , it is intuitive that the first measures will have a high value in the target state determination, since they are the only information the filter has. It is also intuitive that the choice for the gain value will affect the speed of convergence of the filter to the correct estimates of position and velocity. We now seek a way to calculate the "best" (in some sense) gain to be used in the filter at each measurement. That is the work performed by the Kalman filter.

b. The Kalman Filter

Consider a system of order zero (stationary target) from which two measures X_i , where i is the time, are obtained. Knowing that the true position is X_0 , if the measures are corrupted by a Gaussian noise and the errors are independent, the first two measures are:

$$\begin{aligned}
 X_1 &= X_0 + e_1, \\
 \text{and } X_2 &= X_0 + e_2.
 \end{aligned}
 \tag{2.9}$$

Estimating the target position by a linear combination of the measures, the estimate and its error are:

$$\begin{aligned}
 X_e &= aX_1 + bX_2 \\
 \epsilon &= X_e - X_0,
 \end{aligned}
 \tag{2.10}$$

where a and b are the linear coefficients. The criteria chosen to select the "best" estimate are *minimum average error* and *minimum square error*. In that case the equation for a null average error, using the facts that the mean for the errors is null and $E[X_0] = X_0$, is:

$$\begin{aligned}
 E[\epsilon] &= E[aX_1 + bX_2 - X_0] \\
 &= E[a(X_0 + e_1) + b(X_0 + e_2) - X_0] \\
 &= aX_0 + bX_0 - X_0 = 0 \\
 \therefore a + b &= 1.
 \end{aligned}
 \tag{2.11}$$

The equation for the average square error using the result of Equation 2.11 is:

$$\begin{aligned}
 E[\epsilon^2] &= E[(a+b-1)X_0 + ae_1 + be_2]^2 \\
 &= E[(ae_1 + be_2)^2] \quad \text{as } (a+b) = 1
 \end{aligned}
 \tag{2.12}$$

Which can be also written as:

$$\begin{aligned}
 E[\epsilon^2] &= E[a^2 e_1^2 + 2ab e_1 e_2 + b^2 e_2^2] \\
 &= a^2 E[e_1^2] + b^2 E[e_2^2] \\
 &= a^2 \sigma_1^2 + b^2 \sigma_2^2 = a^2 \sigma_1^2 + (1-a)^2 \sigma_2^2 .
 \end{aligned}
 \tag{2.13}$$

After differentiating and equating to zero to find the minimum, the gains and the minimum square error will be:

$$a = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} = \frac{\frac{1}{\sigma_1^2}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}
 \tag{2.14}$$

$$b = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} = \frac{\frac{1}{\sigma_2^2}}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

$$E[\epsilon^2] = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} .
 \tag{2.15}$$

Note that the minimum square error is smaller than the variance for the two measures, which means the variance for the estimate is better than the variance of the two measures. Note also that if

$$\sigma_2 > \sigma_1 , \text{ then } a > b ,
 \tag{2.16}$$

which means that more weight is given to the more precise measure. These results will be applied for the minimum mean square error in a recursive filter that also

combines two values, the measure and the prediction. The best gain for each step is given by Equation 2.14.

$$\alpha = \frac{\sigma_p^2}{\sigma_m^2 + \sigma_p^2} \quad (2.17)$$

where σ_m^2 = variance of the measure

σ_p^2 = variance of the prediction

For a first order system (constant speed) the process of estimation has to be preceded by a prediction (Figure 3 and Equations 5 and 6). The intent now is to calculate α and β following a minimum mean square error criteria of optimization that is

$E[\epsilon_{X_e}^2]$ is minimum $\therefore \alpha$ is the best

$E[\epsilon_{V_{X_e}}^2]$ is minimum $\therefore \beta$ is the best

where $\epsilon_{X_e} = X_e - X$ error in estimated position

$\epsilon_{V_{X_e}} = V_{X_e} - V_X$ error in estimated velocity

V_X = true velocity

X = true position

At the instant of the measurement a prediction operation is performed. The values for the position and the error in predicted position are:

$$\begin{aligned}
X_p(t+\Delta t) &= X_e(t) + V_{X_e} \Delta t \\
\epsilon_{X_p} &= \epsilon_{X_e} + \epsilon_{V_{X_e}} \Delta t \quad .
\end{aligned}
\tag{2.18}$$

The equations for the velocity and velocity error are:

$$\begin{aligned}
V_{X_p}(t+\Delta t) &= V_{X_e}(t) \\
\epsilon_{V_{X_p}} &= \epsilon_{V_{X_e}} \quad .
\end{aligned}
\tag{2.19}$$

The equations for the mean square errors are:

$$\begin{aligned}
E[\epsilon_{X_p}^2] &= E[\epsilon_{X_e}^2 + 2\epsilon_{X_e}\epsilon_{V_{X_e}} + \epsilon_{V_{X_e}}^2] \\
&= \sigma_{X_e}^2 + 2\sigma_{X_e V_{X_e}} + \sigma_{V_{X_e}}^2 \quad .
\end{aligned}
\tag{2.20}$$

$$E[\epsilon_{V_{X_p}}^2] = E[\epsilon_{V_{X_e}}^2] = \sigma_{V_{X_e}}^2 \quad . \tag{2.21}$$

But these are the values for variance of position after prediction and variance of velocity after prediction.

$$\begin{aligned}
\sigma_{X_p}^2 &= \sigma_{X_e}^2 + 2\sigma_{X_e V_{X_e}} + \sigma_{V_{X_e}}^2 \\
\sigma_{V_{X_p}}^2 &= \sigma_{V_{X_e}}^2 \quad .
\end{aligned}
\tag{2.22}$$

At time t a measure is obtained and a new estimate is constructed. The equations for position and variance of position can be derived as [Ref. 1]:

$$X_e = X_p + \alpha(X_m - X_p)$$

$$\sigma_{X_e}^2 = (1-\alpha)\sigma_{X_p}^2 = (1 - \frac{\sigma_{X_p}^2}{\sigma_{X_p}^2 + \sigma_m^2})\sigma_{X_p}^2 = \frac{\sigma_m^2 \sigma_{X_p}^2}{\sigma_{X_p}^2 + \sigma_m^2} \quad (2.23)$$

The equations for velocity and variance of velocity after estimation can be derived as [Ref. 1]:

$$V_{X_e} = V_{X_p} + \beta(X_m - X_p) \quad (2.24)$$

$$E[\epsilon_{V_{X_e}}^2] = \sigma_{V_{X_p}}^2 + 2\beta\sigma_{X_p V_{X_p}}^2 + \beta^2\sigma_m^2 + \beta^2\sigma_{X_p}^2$$

After differentiating the last equation and equating it to zero, the equation for the velocity gain is:

$$\beta = \frac{\sigma_{X_p V_{X_p}}^2}{\sigma_m^2 + \sigma_{X_p}^2} \quad (2.25)$$

So, after the operation of estimation, the variance will be

$$\sigma_{V_{X_e}}^2 = E[\epsilon_{V_{X_e}}^2] = \sigma_{V_{X_p}}^2 - \beta\sigma_{X_p V_{X_p}}^2 \quad (2.26)$$

To complete the algorithm it is necessary to determine the covariances between position and velocity for the operations of prediction and estimation. Those can be derived after some computation as [Ref. 1]:

$$\begin{aligned} \sigma_{X_p V_{X_p}}^2 &= E[\epsilon_{X_p} \epsilon_{V_{X_p}}] = E[(\epsilon_{X_e} + \epsilon_{V_{X_e}}) \epsilon_{V_{X_e}}] = \sigma_{X_e V_{X_e}}^2 + \sigma_{V_{X_e}}^2 \\ \sigma_{X_e V_{X_e}}^2 &= E[\epsilon_{X_e} \epsilon_{V_{X_e}}] = (1 - \alpha)\sigma_{X_p V_{X_p}}^2 \end{aligned} \quad (2.27)$$

The equations described are used sequentially in predictions (or movements) and estimations (or measurements) to update the state vector and

covariance matrix. The system condition is then revised recursively at each time step either by a movement and a measurement or just by a movement when no observation is available.

"All of Kalman's computations can be thought of as manipulations of (multivariate) Normal probability distributions" [Ref. 6]. The attractive feature of the linear plant model is that normality is preserved.

The Kalman filter, in the way it was described in this section, works for targets with constant course and speed. If the target maneuvers after the filter has settled with an optimal gain, the measures will not affect the gain and the filter will not follow the maneuver. To prevent that "disconnection from reality" the realistic deviation of the target from a precisely straight, or other deterministic track is modeled by a random process.

It is important not to confuse the model noise, also called plant noise, with the measure noise. The measure noise appears due to the inaccuracy and uncertainty in the measurement and is simulated by a random number generator. The plant noise is placed in the model to reflect reality and is achieved by the plant noise covariance matrix Q .

To simplify the calculation, the Kalman filter can be represented in matrix notation. The operations performed are the same, but the manipulation is much easier. A primer for the subject is Reference 6, from where the notation will be borrowed.

The process can be divided in two steps, the *measurement step* that corresponds to the prediction, and the *movement step* that corresponds to the estimation. The equations for the steps, omitting the time index, are:

movement step

$$\mu = \Phi\mu + \mu_w \quad (2.28)$$

$$\Sigma = \Phi\Sigma\Phi^T + Q$$

measurement step

$$K = \Sigma H^T (H\Sigma H^T + R)^{-1} \quad (2.29)$$

$$\mu = \mu + K(Z - \mu_v - H\mu)$$

$$\Sigma = (I - KH)\Sigma \quad ,$$

where

Φ is the plant matrix

(μ_w, Q) is the mean and covariance of the plant noise

H is the measurement matrix

(μ_v, R) is the mean and covariance of the measurement noise

(μ, Σ) is the mean and covariance of the state of the system

Z is the measurement vector

K is the Kalman gain matrix

I is the identity matrix .

B. MOTION MODELS

When tracking a target, detection equipment will make observations at intervals of time, which may or may not be equal. Between those measures, the tracker will have to draw inferences about the target's position. For that, the tracker must assume that the target is moving in a certain way, which is called the motion model (Figure 4).

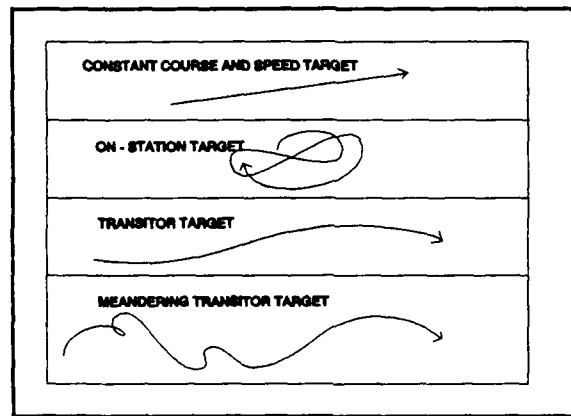


Figure 4 - Motion Model Examples.

The motion model reflects the way the target is expected to behave. It forecasts the position, compares with the observation, and reveals the likely or unlikely track. The simplest motion model is the constant course and speed model, when the tracker assumes the target is moving in a straight line with constant velocity. But many other possibilities exist. For example the GST (Generic Statistical Tracker) uses four motion models [Ref. 4]. The GST keeps track of both the target position (in two dimensions) and velocity. By changing some parameters in the tracker one can obtain the motion models described in Figure 4. As can be deduced, target's position prediction is as good as the coincidence of the motion model used and the target real motion.

Many motion models are not physically realizable by any ship. One model that is physically realizable is the *random tour* where the target moves at constant speed and executes random (uniform from 0 to 2π) course changes at exponentially distributed random times. The random tour can be approximated by a Gauss-Markov process called *Ornstein-Uhlenbeck* process. This approximation is actually very good, under certain assumptions and the process can be used as a motion model with many advantages [Ref. 3].

1. The Ornstein-Uhlenbeck Process²

The Ornstein-Uhlenbeck is a special case of the Generalized Langevin Model which is a model based in linear stochastic differential equations and can be described by the equations:

$$\begin{aligned} dX(t) &= V_X(t)dt \\ dV_X(t) &= -BV_X(t)dt + K(x,t)dt + \zeta dW(t) \end{aligned} \tag{2.30}$$

where $X(t)$ = *X coordinate at time t*
 $V_X(t)$ = *X coordinate of velocity at time t*
 $K(x,t)$ = *an acceleration produced by an external force field*
 $dW(t)$ = *white noise (Wiener differential)*
 ζ = *scale parameter*
 B = *parameter* .

The Langevin model is used to describe the motion of a particle subjected to a frictional damping force with coefficient $B > 0$, an external force field and random shocks described by a white noise. The Ornstein-Uhlenbeck process is defined when there is no external field ($K(x,t)=0$) and it roughly corresponds to an unconstrained random walk without drift.

² This section is an adapted transcription of Reference 2.

The O.U. model becomes:

$$\begin{aligned} dX(t) &= V_X(t) dt \\ dY(t) &= V_Y(t) dt \end{aligned} \tag{2.31}$$

$$\begin{aligned} dV_X(t) &= -B_1 V_X(t) dt + \zeta_1 dW_1(t) \\ dV_Y(t) &= -B_2 V_Y(t) dt + \zeta_2 dW_2(t) \end{aligned} \tag{2.32}$$

where B_1, B_2, ζ_1 and ζ_2 are positive constants
 $dW_1(t)$ and $dW_2(t)$ are white noise .

The O.U. process becomes I.O.U. (Integrated Ornstein-Uhlenbeck) process when Equations 2.31 and 2.32 are integrated to compute the values for position and velocity. Looking at the velocity expression (Equation 2.30) one can see that the velocity component satisfies a stochastic differential equation that reflects deterministic and random influences. The first term decelerates the target at a rate proportional to its velocity, B being the constant of proportionality, and is the deterministic part. Additionally, the target undergoes a rapidly varying random acceleration which is idealized as a white noise. After a time, the first term will decay to zero and the random term will dominate. That means the ability to predict the velocity at a future time will decay asymptotically to zero, with B as the parameter that controls the rate at which the deterministic part becomes dominated by the random component. Although the differential equation characterization of the O.U. process is not hard to solve it will not be derived in this work. The computation developed in this report will use a discretized version [Ref 4]. The equations for the I.O.U. process in matrix notation will appear later in Chapter III.

It is important to understand the exact meaning of the parameters ζ and B of the I.O.U. motion model. The I.O.U. process involves white noise and, in the technical terminology of filtering theory, is non-realizable. No ship of any kind, merchant or warship, that has ever been built can move as required by the O.U. or I.O.U. stochastic differential equations models. By contrast, the Random tour family of models are all physically realizable by actual ships. The approximation of a R.T. by an I.O.U. motion model is actually very good and a correspondence between the parameters can be established. The Random tour is generated by a target moving with constant velocity V and changing course at intervals of time exponentially distributed with mean time between course alterations $1/\lambda$. The courses are chosen independently from a uniform distribution $[0, 2\pi]$. Thus the average number of course changes in $(0, t]$ is λt . It can be shown that the correspondence between the R.T. and I.O.U. parameters is

$$\begin{aligned} B &= \lambda \\ \frac{\zeta^2}{B} &= V^2 \end{aligned} \tag{2.33}$$

so the parameters B and ζ can be described as:

$$\begin{aligned} B &= \text{effective average rate of target course} \\ &\quad \text{change per unit time} \\ \frac{\zeta}{\sqrt{B}} &= \text{RMS target speed} \end{aligned}$$

The I.O.U. process will be used in this work to provide a motion model for targets with random behavior as will be described.

III. MODEL DESCRIPTION

When we talk about target tracking, normally a single filter will not respond perfectly to all situations. We will now describe four filters for three different motion models. These filters will be developed to accomplish our test purpose which is to examine the effects of non-normal outlier-prone observational errors on Kalman filtered tracks. They are not optimal for all situations. The test will be performed with three tracks: Inbound constant course and speed target; Crossing constant course and speed target; and Random Tour target. The fourth filter will be developed to demonstrate the use of two sources of information tracking a single target.

A. DESCRIPTION OF THE ONE SOURCE (SINGLE TARGET) MODELS

1. The Range, Bearing, Radial and Angular Velocity Model (Filter Model 1)

The problem geometry is defined by the observations on range and bearing of a target from a stationary source, positioned at the origin of a Cartesian coordinate system (Figure 5). It seems natural to keep the same orientation for a model. The state vector for that case would have four components; range, bearing, radial and angular velocity and can be represented as:

$$\mu = (\mu_R, \mu_\theta, \mu_{V_R}, \mu_{V_\theta})^T \quad (3.1)$$

This approach tries to keep the measurement as a linear function of the state vector and thus to avoid the use of an extended Kalman filter.

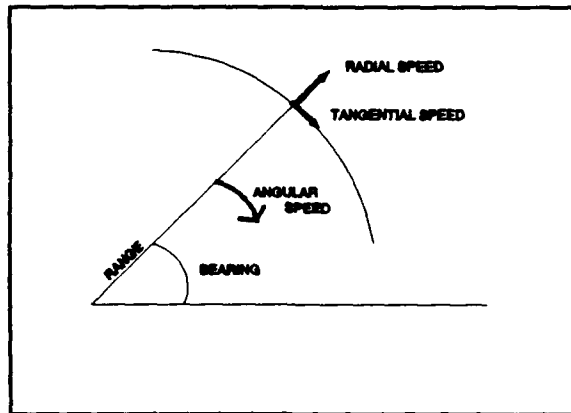


Figure 5 - Problem Geometry.

The equations for the Kalman filter in matrix notation, were described in Chapter II (Equations 2.28 and 2.29). Those equations will be employed to derive the matrix to be used in the model. This model will be used on experiment 1 (inbound CCS target).

a. The movement step

The state vector and the covariance matrix are updated at each movement step. First the Kalman filter "moves" the target in accordance with the plant model, estimating the position just before the next measurement (just like a dead reckoning would do using the last position course and speed). Second it computes the expansion that the time elapsed between steps causes in the target's region of uncertainty. To find the plant model matrix (Φ) we look at the equations of that model:

$$\begin{aligned}
\mu_R(t+\Delta t) &= \mu_R(t) + \mu_{V_R}(t)\Delta t + \text{noise} \\
\mu_\theta(t+\Delta t) &= \mu_\theta(t) + \mu_{V_\theta}(t)\Delta t + \text{noise} \\
\mu_{V_R}(t+\Delta t) &= \mu_{V_R}(t) + \text{noise} \\
\mu_{V_\theta}(t+\Delta t) &= \mu_{V_\theta}(t) + \text{noise} .
\end{aligned}
\tag{3.2}$$

These equations represent a target movement from time (t) to time (t + Δ), using the velocity computed in the previous state and adding a plant noise that will be null in this model. The plant matrix has to satisfy the equations described, which can be summarized as $\mu(t+\Delta) = \Phi \mu(t) + \text{noise}$. The result for the plant matrix is:

$$\Phi = \begin{vmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}
\tag{3.3}$$

The plant noise covariance matrix Q, is defined for this model as a null matrix.

b. Measurement Step

The measurement step will first compute the Kalman gain matrix K. The Kalman gain determines how seriously the new measured value will be taken and how extensively the estimate will "bounce"; a small K makes the filter lethargic. The state vector and the state covariance matrix are then updated. The first gives the best estimate of the target at the moment and the second shrinks the area of uncertainty. The state covariance matrix reflects the uncertainty of the state vector components

and with the state vector, condenses all the information in the system state. The measurement matrix H relates the 2 measurements to the state vector by the equation

$$Z = H X + V \quad (3.4)$$

where V is the measurement noise. In our model H will be a 2×4 matrix of the form

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (3.5)$$

2. The (X, Y, V_x, V_y) Simple Model (Filter Model 2)

A model using the Cartesian coordinate system has some general advantages over the previous model such as: the transition to a model with two sensors is sometimes less complicated than for one in polar coordinates; a target with constant course and speed has constant velocities in X and Y ; the process is independent for the coordinates X and Y , so the computation can be easily divided for parallel processing. The only problem is to keep the observational variables (range and bearing) a linear combination of the state vector (X, Y, V_x, V_y) . A conversion of coordinates from polar coordinates to Cartesian coordinates is used prior to the application of the data into the filter, making the filter believe that everything is in coordinates X and Y [Ref. 7]. The equations that relate range and bearing with the coordinates X and Y are well known to be:

$$\begin{aligned} X &= R \sin(\theta) \\ Y &= R \cos(\theta) \end{aligned} \quad (3.6)$$

"On account of the non-linear transformation (Equation 3.9), the measurement errors on the Cartesian coordinates are non-Gaussian distributed and the optimal filter would be non-linear" [Ref. 7]. In order to avoid this difficulty, assume as a reasonable hypothesis that the errors in polar coordinates are small compared with the true target coordinates. Under this assumption the Cartesian errors are obtained by differentiation of Equation 3.9 and the relationship between the errors in both coordinates system is linear and the Gaussian probability distribution is maintained. It is also necessary to convert the variances in range and bearing to variances in X and Y. The formulas are:

$$\begin{aligned} \sigma_X^2 &= \sin^2(\theta) \sigma_R^2 + R^2 \cos^2(\theta) \sigma_\theta^2 \\ \sigma_Y^2 &= \cos^2(\theta) \sigma_R^2 + R^2 \sin^2(\theta) \sigma_\theta^2 \\ \sigma_{XY} &= (\sigma_R^2 - R^2 \sigma_\theta^2) \sin\theta \cos\theta \end{aligned} \quad (3.7)$$

A model using those transformations can be easily derived in a way similar to that done for the first model.

a. Movement Step

The state vector is composed of the coordinates X and Y and the respective velocities. The notation used is:

$$\mu = (\mu_X, \mu_Y, \mu_{V_X}, \mu_{V_Y})^T \quad (3.8)$$

To define the plant matrix it is necessary to look at the equations of the plant model

$$\begin{aligned}
\mu_x(t+\Delta t) &= \mu_x(t) + \mu_{v_x}(t)\Delta t + \text{noise} \\
\mu_y(t+\Delta t) &= \mu_y(t) + \mu_{v_y}(t)\Delta t + \text{noise} \\
\mu_{v_x}(t+\Delta t) &= \mu_{v_x}(t) + \text{noise} \\
\mu_{v_y}(t+\Delta t) &= \mu_{v_y}(t) + \text{noise} .
\end{aligned}
\tag{3.9}$$

A plant matrix Φ equal to that of the previous model (Equation 3.3) will satisfy these equations. The plant noise for the purpose of the test (CCS target) will be null.

b. Measurement Step

The measurement covariance matrix R uses the converted variances from range and bearing to X and Y (Equation 3.7). As the matrix H simply extracts the first two components of the state vector and the conversion from the measurement in range and bearing to X and Y is prior to the use of the data, the matrix H is the same for all models (Equation 3.4).

3. The (X, Y, V_x, V_y) Model Using I.O.U. Process (Filter Model 3).

As mentioned in Chapter II, the I.O.U. motion model may be a good approximation to the Random Tour motion model, still allowing one to remain within the class of Gaussian approximations to target motion. The modifications from the previous model are only in the movement step, so we will not explain the measurement step.

a. Movement Step

To derive the plant model matrix (Φ) for the model we first look at the equations that describe the motion model:

$$\begin{aligned}
X(t+\Delta t) &= X(t) + \frac{1}{B\Delta t}(1-e^{-B\Delta t}) V_X \Delta t \\
Y(t+\Delta t) &= Y(t) + \frac{1}{B\Delta t}(1-e^{-B\Delta t}) V_Y \Delta t \\
V_X(t+\Delta t) &= e^{-B\Delta t} V_X(t) \\
V_Y(t+\Delta t) &= e^{-B\Delta t} V_Y(t) .
\end{aligned} \tag{3.10}$$

So the plant matrix is:

$$\Phi = \begin{vmatrix} 1 & 0 & b1 & 0 \\ 0 & 1 & 0 & b1 \\ 0 & 0 & b2 & 0 \\ 0 & 0 & 0 & b2 \end{vmatrix} \tag{3.11}$$

where

$$\begin{aligned}
b1 &= \frac{1}{B}(1-e^{-B\Delta t}) \\
b2 &= e^{-B\Delta t}
\end{aligned}$$

The plant noise to be injected in the system has mean zero and covariance matrix Q defined as [Ref. 4]:

$$Q = \begin{vmatrix} c1 & 0 & c2 & 0 \\ 0 & c1 & 0 & c2 \\ c2 & 0 & c3 & 0 \\ 0 & c2 & 0 & c3 \end{vmatrix} \tag{3.15}$$

where

$$\begin{aligned}
c1 &= \frac{1}{2} \left(\frac{\zeta}{B} \right)^2 [2\Delta t - \left(\frac{1}{B} \right) (3 - 4e^{-B\Delta t} + e^{-2B\Delta t})] \\
c2 &= \frac{1}{2} \left[\left(\frac{\zeta}{B} \right) (1 - e^{-B\Delta t}) \right]^2 \\
c3 &= \frac{1}{2} \left(\frac{\zeta^2}{B} \right) (1 - e^{-2B\Delta t}) .
\end{aligned} \tag{3.16}$$

A Fortran code for this model is presented in Appendix A.

B. DESCRIPTION OF THE 2 SOURCES (SINGLE TARGET) MODEL.

Considering the same type of sensor, two sources of information will normally give more information than just one. When more than one source is available and they are separated in space, the necessary link and information fusion may degrade the system reliability. Sometimes the second source is not available at all and the problem has to be solved for just one platform, optimizing the information available. "The combination of 2 sources of information can be realized in a *track selection fusion*, in which the tracker with the best solution is selected and no real fusion is computed, or in a *state vector fusion*, in which the associated state vectors are combined in a linear estimator to derive a central-level state estimate" [Ref. 8]. The estimator in this case must consider sensor-specific parameters and can be implemented in a fashion similar to the Kalman filter to form a minimum mean square estimate. Another possibility is *measurement fusion* that uses the measures to compute a central estimate of target state. In this case we have to consider if the measures are synchronous or not, as that changes the way we combine them before the application into the filter.

In this work a *state vector fusion* using a Kalman filter with the I.O.U. motion model is implemented. In this case only one fused state vector and fused covariance

matrix is employed to represent the system. The problem geometry is shown in Figure 6. The sources are assumed to have the same orientation (positive X axis is to the right). Note that the error ellipses for each individual source's observations don't have the same orientation, so the measurement covariance matrix R is different for each source. However, conversion of the variances from range and bearing to X and Y (Equation 3.10) before using the data in the filter, allows the matrix R , although reflecting measures with different orientations, can be used by both sources with no distinction to update a unique fused state vector. The good part is that the Kalman filter will not just combine measures with different orientation but will also take into account the precision reflected in the values of the matrix R and will give more weight to the better measure.

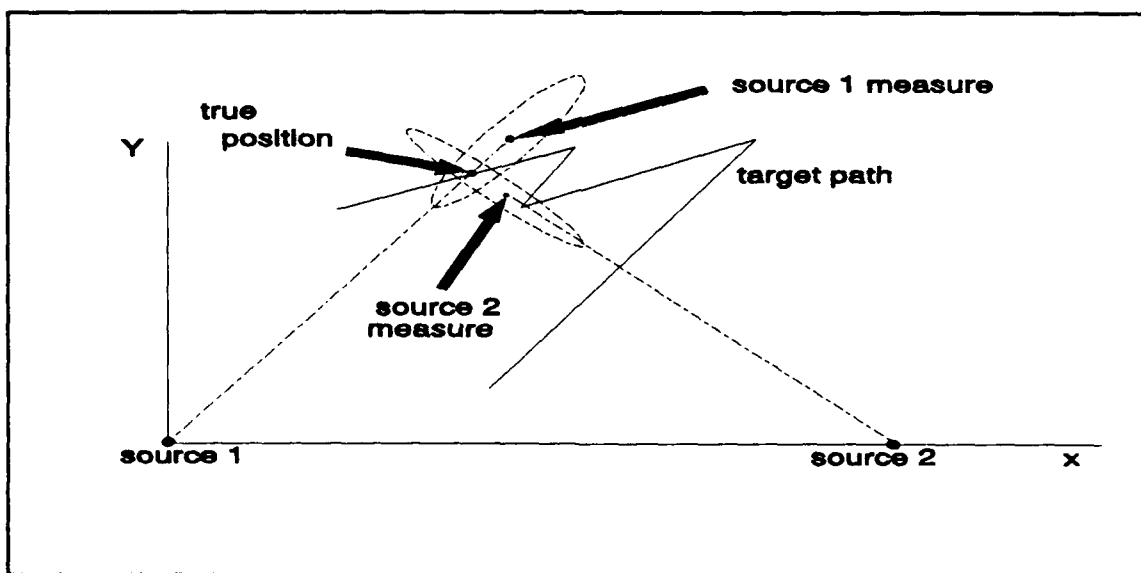


Figure 6 - Two sources problem geometry

IV - MODEL EVALUATION

A. TARGET MOTION GENERATOR

Evaluation of a tracker requires a target motion generator to create possible paths that a target could produce. Noise must be added to simulate the observed track. The observations arrive as a sequence of bearings and ranges of the target, so after computing the X and Y coordinates of each motion step, we calculate the range and bearing from the tracker unit and then add the noise. This is an attempt to represent actual measured physical conditions, specifically by assuming a range error with mean zero and standard deviation equal to a percentage of the current target real distance, and a normal bearing error with mean zero and constant standard deviation. A Fortran code of the motion generator is presented in Appendix B, and a more detailed description follows.

1. Target Measurement Error

The target motion generator has 2 different measurement noises that can be used, a Normal noise and a non-Normal, outlier-prone noise simulated by *sculpturing* a Normal distribution. The normal noise is generated using the method of Box-Muller [Ref. 9]. The bearing standard deviation is 0.01 radian, and that for the range is a constant percentage of the actual target range (input for the program). Those figures can, of course, be modified to any desired value.

A sculptured noise [Ref. 10] is a modification that stretches the tails of the normal noise to any desired value of kurtosis³. This device models the situation when the noise is not normal but still symmetric. It represents the occurrence of gross errors or outliers. The process of distributional sculpturing, described in Reference 6, suggests two shaping functions for stretching symmetric distributions:

$$\begin{aligned} (i) \quad S(Z) &= 1 + hZ^2 \quad h > 0 \\ (ii) \quad S(Z) &= e^{hZ^2} \quad h > 0 \end{aligned} \quad (4.1)$$

If Z is a random variable normally distributed with mean 0 and standard deviation σ , then $Y = Z \times S(Z)$ resembles the distribution of Z for small values, but lengthens the tails of the distribution for large Z .

To obtain the desired kurtosis in the sculptured distribution it is necessary to analyze the moments (particularly the second and fourth) of that distribution. They are represented (Equations 4.2 and 4.3) in terms of those for the original Z for each transformation.

$$\begin{aligned} (i) \quad Y &= Z(1+hZ^2) \\ m_1(Y) &= m_1(Z) + h m_3(Z) = 0 \\ m_2(Y) &= m_2(Z) + 2h m_4(Z) + h^2 m_6(Z) \\ m_4(Y) &= m_4(Z) + 4h m_6(Z) + 6h^2 m_8(Z) + 4h^3 m_{10}(Z) + h^4 m_{12}(Z) \end{aligned} \quad (4.2)$$

³ It is also possible to sculpture for skewness, but this is not done in this work.

$$(ii) \quad Y = Ze^{hZ^2} \quad (4.3)$$

$$m_2(Y) = \frac{\sigma^2}{(1-4h\sigma^2)^{3/2}} \quad h < \frac{1}{4\sigma^2} \quad (4.4)$$

$$m_4(Y) = \frac{3\sigma^4}{(1-8h\sigma^2)^{5/2}} \quad h < \frac{1}{8\sigma^2} \quad , \quad (4.5)$$

Where σ^2 is the variance of Z [Ref. 10]. Those formulas are applicable to any Normal distribution. If h exceeds the indicated limits (Equations 4.4 and 4.5) the moments become infinite; even so the models may be of interest. The moments for the Gaussian distribution with mean 0 and standard deviation σ can be found by expanding both sides of the equality

$$E[e^{\theta Z}] = e^{\frac{1}{2}\theta^2\sigma^2} \quad (4.6)$$

and comparing the terms with same degree in θ . It can be seen that the odd moments are null and the even moments are given by the formula

$$E[Z^n] = \frac{n!}{(\frac{n}{2})! 2^{\frac{n}{2}}} \sigma^2 \quad (4.7)$$

The tail extension can be measured by the kurtosis, obtained by dividing the fourth moment by the square of the second moment, and then subtracting the constant factor 3. Calculation of the fourth moment for transformation (i) needs all the even moments up to the twelfth of the original distribution (Normal in our case).

$$\begin{aligned}
E[Z^2] &= \sigma^2 \\
E[Z^4] &= 3\sigma^4 \\
E[Z^6] &= 15\sigma^6 \\
E[Z^8] &= 105\sigma^8 \\
E[Z^{10}] &= 945\sigma^{10} \\
E[Z^{12}] &= 10395\sigma^{12}
\end{aligned} \tag{4.8}$$

The purpose of this work is to compare the behavior of the tracker when using an observational error noise that is Normal with mean 0 and standard deviation σ , with the performance when the error noise is sculptured with the same mean and standard deviation but with a given kurtosis. The procedure is to start with a normal distribution that, after the transformation, will be sculptured with standard deviation of σ and a given kurtosis. An algorithm that carries this on is as follows:

1. Generate a random variable Z that is $N(0,1)$.
2. Pick an appropriate value of sculpturing parameter h .
3. Apply the transformation to the random variable Z to get Y . Divide by the transformed standard deviation and then multiply by the desired standard deviation σ_R .

$$Z_T = \frac{Y}{SD[Y]} \sigma_R \tag{4.9}$$

After those steps the variance of Z_T is σ_R and the kurtosis is

$$\gamma_2(Z_T) = \frac{E[Z_T^4]}{E[Z_T^2]^2} - 3 \tag{4.10}$$

The attractive feature of the outlined solution is that the original distribution is $N(0,1)$, simplifying the calculation of the moments for the sculptured distribution.

Applying the solution to the first transformation:

1. Define a $N(0,1)$ random variable Z
2. Define a random variable W and a random variable Z_T as

$$W = \frac{Z(1+hZ^2)}{\sqrt{\text{VAR}[Z(1+hZ^2)]}} \quad (4.11)$$

$$Z_T = W \sigma_R$$

At this point the variance of Z_T is σ_R^2 and the kurtosis is:

$$\begin{aligned} \gamma_2[Z_T] &= \frac{\sigma_R^4 E[W^4]}{(\sigma_R^2 E[W^2])^2} - 3 = \frac{E[W^4]}{1} - 3 \\ &= E \left[\left(\frac{Z(1+hZ^2)}{\sqrt{\text{VAR}[Z(1+hZ^2)]}} \right)^4 \right] - 3 \end{aligned} \quad (4.12)$$

After some computation the Kurtosis can be expressed in terms of h as:

$$\gamma_2[Z_T] = \frac{10395h^4 + 4(945)h^3 + 6(105)h^2 + 4(15)h + 3}{(15h^2 + 6h + 1)} - 3 \quad (4.13)$$

The appropriate solution for Equation 4.13 will give the correct value for h to be used in Equation 4.11. The results of solving the equation numerically for different values of kurtosis are presented in Table 1.

TABLE 1 - SOLUTIONS FOR EQUATION 4.12

Kurtosis	h
1	0.0347
2	0.0622
3	0.0867
4	0.1097
5	0.1319
6	0.1537
7	0.1755

Similarly, solutions for the second transformation can be found. The values for h, for both transformations, were confirmed by a simulation using APL2/32.

2. Target Motion Pattern.

The target motion generator has 4 different motion patterns, each with a particular characteristic to test. All 4 patterns have the same output to the file MOTIOx RES (the 'x' is the motion model number) containing 9 variables described below:

- TIME - Simulation time (hours).
- X - Target x true coordinate (nautical miles).
- Y - Target y true coordinate (nautical miles).

- RANGE - Target true range (nautical miles).
- BEARI - Target true bearing (degrees with 000 being the positive x axis).
- XN - Target x coordinate with normal error (nautical miles).
- YN - Target y coordinate with normal error (nautical miles).
- RANGEN - Target range with normal error (nautical miles).
- BEARIN - Target bearing with normal error (as in BEARI).

Only the 2 patterns that will be used in the experiments will be described.

a. Motion Pattern 1 - Constant Course and Speed Target

This pattern accepts as input the target initial position (coordinates X and Y in nautical miles), course and speed. It then updates the target position at each time step by using the Equation 4.14. The result is a straight path with the course computed counterclockwise from the positive X axis and constant speed.

$$\begin{aligned} X(t+\Delta t) &= X(t) + V_x \Delta t \\ Y(t+\Delta t) &= Y(t) + V_y \Delta t \end{aligned} \quad (4.14)$$

Where V_x and V_y are the velocities in X and Y respectively.

b. Motion Pattern 4 - Random Tour Target

This motion model was described in Chapter II. The inputs are the target initial coordinates, speed, and the parameter λ for the exponential distribution (the input is $1/\lambda$).

B. EVALUATION.

To gain an insight into the problem, which is to examine the effect of non-normal outlier-prone observational error distributions in the Kalman filter track, we will perform 3 experiments using a 2 level factorial design [Ref. 7]. In a 2 level factorial design a set of orthogonal dummy variables is created and a linear regression is performed. Each dummy variable has 2 possible values (+1 or -1) that correspond to the high and low level of the factors. The model used for the linear regression of the dummy variables on the response variable being analyzed is

$$RESP = \eta_0 + \eta_1 d_1 + \eta_2 d_2 + \eta_3 d_3 + \eta_4 d_1 d_2 + \eta_5 d_1 d_3 + \eta_6 d_2 d_3 + \eta_7 d_1 d_2 d_3$$

WHERE

RESP is the response variable, (4.15)

d_i are the dummy variables for K , SD and ΔT and

η_i are the regression coefficients.

The computed regression coefficients will possibly indicate how the response variable will behave when we change the factors from low to high level (for further information see Box, Hunter and Hunter [Ref. 11]). Three factors will be used in the design:

1. Kurtosis of the range error distribution.
2. Standard deviation of the range error distribution.
3. Time interval between observations.

One experiment consists of 2^3 combinations of factors and levels. A certain number of runs (500 or 1000) is performed for each combination. Each run consists of a sequence of observations. For each observation, the squared distance between the true position and the estimated position, D^2 , is computed. The distances for one

complete run are then averaged, and that will be used as the experiment response variable. So, the response variable is the expected value for D^2 ($E[D^2]$) for one run that will be called ED^2 . The sample variance for the response variable is also computed in the simulations. The levels for the factors are presented in Table 2.

TABLE 2 - FACTORS AND LEVELS

FACTOR	- LEVEL	+ LEVEL
KURTOSIS (K)	0 (Normal)	2 (3 for the first experiment)
SD	5% OF RANGE	10% OF RANGE
TIME INTERVAL (ΔT)	0.005	0.01

The standard deviation for bearing errors is constant and has the value 0.01 Radians.

1. Experiment 1: Inbound Target

An inbound target is running at constant course 225° (000° is the positive X axis) and constant speed 10 and is tracked by filter model 1. A run starts at 60 miles from the source and stops at 40 miles. A number of these runs is performed until we reach a total of 200,000 observations for each combination. That means 500 runs of 400 observations each for $\Delta T = .005$ and 1000 runs of 200 observations each for $\Delta T = .01$. The state vector is initialized for each run with the first observation and null velocities. The covariance matrix is also initialized at each run with the values:

$$\Sigma = \begin{vmatrix} 100 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0.1 \end{vmatrix} \quad (4.16)$$

The results are presented in Table 3. Note that the response variable has small values, considering that the mean distance of the target from the source is 50 miles. As we are using a null plant noise covariance matrix, after a short while the filter tracks with precision and gives a very small weight to the observations.

TABLE 3 - RESULTS FOR EXPERIMENT 1

K	SD	ΔT	ED ²	VAR[ED ²]
-	-	-	0.284	0.0098
+	-	-	0.282	0.0117
-	+	-	1.005	0.1433
+	+	-	0.996	0.1835
-	-	+	0.503	0.308
+	-	+	0.498	0.377
-	+	+	1.753	0.436
+	+	+	1.741	0.5901

The pooled sample variance is defined as [Ref. 7]:

$$S^2 = \frac{\Gamma_1 S_1^2 + \Gamma_2 S_2^2 + \Gamma_3 S_3^2 + \dots + \Gamma_g S_g^2}{\Gamma_1 + \Gamma_2 + \Gamma_3 + \dots + \Gamma_g} \quad (4.17)$$

$$\text{with } \Gamma = \Gamma_1 + \Gamma_2 + \dots + \Gamma_g \quad df \quad (4.18)$$

where $\Gamma_i = df.$ for the estimate of variance for set of runs i
 $= \# \text{ of runs} - 1$

After computation the pooled variance is 0.18045. Since each main effect is a subtraction of two averages, with each average containing eight observations, the variance of each effect is:

$$\begin{aligned} V_{\text{effect}} &= V(\bar{X}_- - \bar{X}_.) = \left(\frac{1}{8} + \frac{1}{8}\right) S^2 = \frac{S^2}{4} \\ &= 0.045 \end{aligned} \quad (4.16)$$

The main effects and interactions were computed and the results are presented in Table 4. The values found in Table 4 can be related to the regression coefficients (Equation 4.15) after the inclusion of a scale factor [Ref. 11].

The conclusions to be extracted from this experiment are: there are no significant (more than 2 STD. ERROR) interaction effects, so the main effects can be considered separately; for the levels and variations between levels used, the standard deviation of the measured errors affects the track significantly (as we should expect), and the effect of kurtosis and interval between observations are not significant. As will be seen in the third experiment (I.O.U. model) the levels used affect the result of the experiment.

TABLE 4 - EFFECTS AND INTERACTIONS (EXPERIMENT 1)

EFFECT	ESTIMATE	STD. ERROR
K	-0.0072	0.21
SD	0.9820	0.21
ΔT	0.4818	0.21
K SD	-0.0034	0.21
K ΔT	-0.0019	0.21
SD ΔT	0.2645	0.21
K SD ΔT	-0.0004	0.21

2. Experiment 2: Crossing Target

A crossing target is running at constant course 315° and constant speed 10 and is tracked by filter model 2 (X,Y,V_x,V_y simple model). A run starts at coordinates X = 26 miles and Y = 42 miles, stopping 2 hours later. A total of 500 runs of 400 observations each for $\Delta T=0.005$ and 1000 runs of 200 observations each for $\Delta T=0.01$ is performed. The state vector is initialized, for each run, with the first observation converted by Equation 3.6 and null velocities. The covariance matrix is also initialized for each run with the values:

$$\Sigma = \begin{vmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{vmatrix} \quad (4.17)$$

The results are presented on Table 5.

TABLE 5 - RESULTS FOR EXPERIMENT 2

K	SD	ΔT	ED ²	VAR[ED ²]
-	-	-	0.272	0.0104
+	-	-	0.284	0.0126
-	+	-	1.736	0.2206
+	+	-	2.000	0.56
-	-	+	0.448	0.033
+	-	+	0.467	0.0423
-	+	+	2.325	0.723
+	+	+	2.565	1.031

The response variable values are also small, but this comes as no surprise since this filter is also optimal for a CCS target. After computation the pooled variance for the response variable is **0.34** and the variance of each effect is **0.086**. The main effects and interactions the results are presented in Table 6. The conclusions to be extracted from this trial are the same as for the previous experiment. A few number

of runs high value of ED^2 were observed on the eighth combination, signifying that the track was bad on those runs.

TABLE 6 - EFFECTS AND INTERACTIONS (EXPERIMENT 2)

EFFECT	ESTIMATE	STD. ERROR
K	0.13	0.29
SD	1.79	0.29
ΔT	0.37	0.29
K SD	0.12	0.29
K ΔT	-0.004	0.29
SD ΔT	0.19	0.29
K SD ΔT	-0.008	0.29

We repeated the experiment and counted the number of runs with ED^2 greater than 7.5 (heuristically assumed as a threshold to consider that the track was lost) and none of the runs in all sets of runs reached that value.

3. Experiment 3: Random Tour Target

a. Single Source Model

A target performing a Random Tour starts movement at coordinates $X=39$ and $Y=39$ with constant speed 10 and is tracked by filter model 3 (X, Y, V_x, V_y I.O.U. model). A set of 1000 runs with 200 observations each is performed for each combination of factors and levels. The state vector and covariance matrix are initialized

as in experiment 2. The number of runs with ED^2 greater than 7.5 are computed. The results showed that for combination number 3 (line 3 of Table 7) the tracker lost track 4 times (out of 1000). The track was also lost for combination 4 (42 times), for combination 7 (7 times) and for combination 8 (80 times). The variances for combinations 4 and 8 (consequently the pooled variance) were so big that significant results could not be extracted. We eliminated those runs where the track was lost and performed the experiment again until 1000 good runs were reached⁴. The results are presented in Table 7.

TABLE 7 - RESULTS FOR EXPERIMENT 3 (ONE SOURCE)

K	SD	ΔT	ED^2	$VAR[ED^2]$
-	-	-	0.571	0.037
+	-	-	0.593	0.052
-	+	-	2.112	0.704
+	+	-	2.356	1.036
-	-	+	0.838	0.059
+	-	+	0.873	0.079
-	+	+	2.88	1.011
+	+	+	3.113	1.291

⁴ Although this procedure is not easily statistically interpretable, it was used for the sake of comparison with the other experiments (bad runs will be considered later).

The pooled variance for the response variable is **0.53** and the variance of each effect is **0.1315**. Results of computation of the main effects and interactions are presented in Table 8. The standard deviation of range error is the only effect that is significant, at the levels and level variations used.

TABLE 8 - EFFECTS AND INTERACTIONS (EXPERIMENT 3, ONE SOURCE)

EFFECT	ESTIMATE	STD. ERROR
K	0.13	0.36
SD	1.89	0.36
ΔT	0.52	0.36
K SD	0.10	0.36
K ΔT	-0.004	0.36
SD ΔT	0.245	0.36
K SD ΔT	-0.007	0.36

The experiments gave insight into the problem. We discovered that the tracker is losing track in a number of runs for the sets of runs that have high levels of K and SD. To quantify the percentage of runs in which the track is lost, we decided to run two more simulations; one at 5% and other at 10% of SD. The program will now vary the kurtosis from 0 to 5, counting the number of tracks lost in 3 categories. The interval between observations is fixed at 0.01 and the bad runs will be considered in the computations. For 5% SD no single run lost track for all values of kurtosis.

For 10% SD the results are summarized in the Table 9. The fields **LOST 1**, **LOST 2** and **LOST 3** are the number of runs ,out of 1000 runs, with ED^2 between 7.5 and 10, 10 and 50 and greater than 50, respectively. The field EDN^2 is the average squared difference between the measured and the true positions.

TABLE 9 - EFFECT OF KURTOSIS VARIATION (ONE SOURCE)

K	0	1	2	3	4	5
LOST 1	7	27	41	51	41	41
LOST 2	0	7	29	50	80	94
LOST 3	0	0	1	8	14	24
ED^2	2.92	3.24	3.87	5.21	7.39	9.79
EDN^2	31.37	31.41	31.45	31.48	31.50	31.53
$VAR[ED^2]$	1.12	1.93	10.59	119.6	501.58	1088.95

As we increase kurtosis the following happens:

1. The values for ED^2 are not increasing much (for small values of SD they even decrease), and in the worst case we are still improving when considering the estimated position compared with the measured position (in average values).

2. What is not apparent in the average values presented is that we will lose track in some runs, and that the number of tracks lost increases rapidly with the kurtosis.

3. The variances are growing very fast as the number of tracks lost increase. This is the reason why we could not get a reasonable pooled variance to compare with the mean effects when the levels of kurtosis and SD were high.

4. The values for EDN^2 are increasing very slowly.

b. Two Sources Model

We now want to access the impact of non-Normal errors on the model that uses two sources of information by observing how the results obtained in the previous section will change. A simulation similar to the last one performed on the previous section was executed using the 2 sources model described in Chapter III. Another source was positioned at coordinates $X = 78$ and $Y = 0$, and alternate observations are made (each source performs an observation at .01 hour interval). Observations from both sources had the same range error distribution. The results are presented in Table 10. The inclusion of an additional source of information does not prevent the tracker from losing track, but there is a large reduction on the number of tracks lost and a considerable improvement on the track quality.

TABLE 10 - EFFECTS OF KURTOSIS VARIATION (TWO SOURCES)

K	0	1	2	3	4	5
LOST 1	0	0	1	2	2	4
LOST 2	0	0	0	2	5	8
LOST 3	0	0	0	0	1	4
ED²	0.68	0.73	0.79	0.89	1.1	1.58
EDN²	31.33	31.37	31.39	31.42	31.44	31.45
VAR[ED²]	0.07	0.11	0.20	0.65	5.39	38.39

V - CONCLUSIONS AND RECOMMENDATIONS.

The conclusions presented in this section reflect the results of the experiments performed in Chapter IV. The values obtained and the conclusions extracted are for the models and levels used.

There is no significant change in the response variable when we switch from the lower to the higher level of kurtosis, keeping constant the other factors. For small values of range error standard deviation (less than 5%), the effect of tail extension in the error distribution is not noticeable in all three models. For larger values of range error standard deviation, as we increase the value of kurtosis the trackers will start to lose track (threshold of 7.5) at the percentages presented on Table 11 for filter model 3. In an attempt to visualize the reasons why the tracks were lost, the seeds for the bad runs were recorded and some of the tracks were reproduced and plotted in GRAFSTAT. A number of plots showed that the first observation was an outlier and the tracker took a long while to recognize this fact. No other apparent reason for track loss (other than bad observations caused by the long tail error distribution) was found.

TABLE 11 - PERCENTAGE OF TRACKS LOST

KURTOSIS	PERCENTAGE OF
	RUNS LOST
1	3.4 %
2	7.1 %
3	10.9 %
4	13.5 %
5	15.9 %

A. SUMMARY

Kalman filtering is an extremely broad subject. The models presented can be embellished to account for multiple targets, false echoes and clutter, targets with random acceleration and many other issues that appear in the real world . The two (and multiple) sources model is another area for research. Comparison between different fusion methodologies: *track selection fusion*, *state vector fusion*, and *measurement fusion* models to find the best solution for an issue is certainly a challenging and practical area for further study.

APPENDIX A - FORTRAN PROGRAM FOR FILTER MODEL 3

PROGRAM KALXY3

```

*****
* THIS PROGRAM SIMULATES THE OPERATION OF A KALMAN FILTER
* USING THE COORDINATES X AND Y AND THE I.O.U. PROCESS
* AS A MOTION MODEL.
*****
*
***** DESCRIPTION FOR SOME VARIABLES *****
* BETA  PARAMETER IN THE IOU PROCESS
* B1,B2,C VARIABLES DEFINED IN THE IOU PROCESS
* C1,C2,C3 VARIABLES DEFINED IN THE IOU PROCESS
* DELTAT INTERVAL TIME BETWEEN OBSERVATIONS (HOURS)
* I,J   AUXILIARY VARIABLE
* LL,CC AUXILIARY VARIABLE
* ID    IDENTITY MATRIX (4X4)
* K     KALMAN GAIN MATRIX (4X2)
* MU    STATE ESTIMATE (4X1)
* R     COVARIANCE MATRIX OF THE MEASUREMENT NOISE (2X2)
* SIGMA PARAMETER IN THE IOU PROCESS
* SIG   COVARIANCE MATRIX ESTIMATE (4X4)
* SIZE  NUMBER OF POINTS TO BE GENERATED IN THE SIMULATION
* TIME  SIMULATION TIME (HOURS)
* Z     OBSERVATION (2X1)
*
*****
*
  INTEGER SIZE,I,J,LL,CC
  REAL BETA,B1,B2,C1,C2,C3,C
  REAL DDAR(2,2),DELTAT,H(2,4),HT(4,2),HSIGHT(2,2),HMU(2,1)
  REAL AUXSIG(4,4),Q(4,4),ID(4,4),IDMIKH(4,4),SCX,SCY
  REAL SIGMA,SIGHT(4,2),SIG(4,4),SIGPLUS(4,4),SCOREX,SCOREY
  REAL K(4,2),KH(4,4),KZMIHMU(4,1),RANGE,BEAR1
  REAL MU(4,1),MUPLUS(4,1),PERC,PHI(4,4),PHIT(4,4),PHISIG(4,4)
  REAL TIME2,XN,YN,Z(2,1),ZMIHMU(2,1),R(2,2),RADD(2,2)
* INITIALIZATION
  PARAMETER (PI = 3.141593)
  DATA H/1.,0.,0.,1.,4*0./, ID/1.,4*0.,1.,4*0.,1.,4*0.,1./
  SIZE = 200
  DELTAT = .01
  SIGMA = 10. * 2. **.5
  BETA = 2.

```



```

PERC = .30
DO 120 LL=1,4
  DO 130 CC=1,4
    PHI(LL,CC) = 0.
    Q(LL,CC) = 0.
    IF (LL.EQ. CC) THEN
      PHI(LL,CC) = 1.
    ENDIF
130  CONTINUE
120  CONTINUE
  PHI(3,3) = EXP(-BETA * DELTAT)
  PHI(4,4) = PHI(3,3)
  PHI(1,3) = (1 - PHI(3,3)) / BETA
  PHI(2,4) = PHI(1,3)
  Q(1,1) = .5 * ((SIGMA / BETA)**2) * (2.*DELTAT - ((3. - 4. *
C    PHI(3,3) + EXP(-2*BETA*DELTAT)) / BETA))
  Q(2,2) = Q(1,1)
  Q(1,3) = .5 * ((SIGMA/BETA) * (1-PHI(3,3))) **2
  Q(2,4) = Q(1,3)
  Q(3,1) = Q(1,3)
  Q(4,2) = Q(1,3)
  Q(3,3) = .5 * (SIGMA**2 / BETA) * (1- EXP(-2* BETA * DELTAT))
  Q(4,4) = Q(3,3)
  OPEN(UNIT=12,FILE='D:\APL2\MOTIO4.RES',STATUS='OLD')
*  OPEN(UNIT=11,FILE='D:\APL2\KALXY3.RES',STATUS='NEW')
  MU(3,1) = 0.
  MU(4,1) = 0.
  SCOREX = 0.
  SCX = 0.
  SCOREY = 0.
  SCY = 0.
  DO 170 LL=1,4
    DO 180 CC=1,4
      SIG(LL,CC) = 0.
180  CONTINUE
170  CONTINUE
    SIG(1,1) = 100.
    SIG(2,2) = 100.
    SIG(3,3) = 50.
    SIG(4,4) = 50.
    DO 200 I=1,SIZE
***** MEASUREMENT STEP *****
*  WRITE(11,*)
*  WRITE(11,*)'MEASUREMENT STEP ',I
  READ(12,110) TIME2,X,Y,XN,YN,RANGE,BEARI
110  FORMAT(1X,F5.2,2(1X,F6.2),19X,2(1X,F8.4),1X,F10.6,1X,F11.6)
  BEARI = BEARI * PI /180.

```

```

*** TRANSFORMATION OF COORDINATES
  Z(1,1) = RANGE * COS(BEARI)
  Z(2,1) = RANGE * SIN(BEARI)
  R(1,1) = (SIN(BEARI)* PERC * RANGE)**2 +
C      (RANGE * COS(BEARI) * .01)**2
  R(2,2) = (COS(BEARI)* PERC * RANGE)**2 +
C      (RANGE * SIN(BEARI) * .01)**2
  R(1,2) = ((PERC* RANGE )**2 - (RANGE* .01)**2)
C      * SIN(BEARI) * COS(BEARI)
  R(2,1) = R(1,2)
*** INITIAL STATE VECTOR
  IF(I.EQ.1) THEN
    MU(1,1) = Z(1,1)
    MU(2,1) = Z(2,1)
  ENDIF
*** COMPUTATION OF THE KALMAN GAIN "K"
  CALL TRANSP(H,2,4,HT)
  CALL MATMUX(SIGHT,SIG,4,4,HT,4,2)
  CALL MATMUX(HSIGHT,H,2,4,SIGHT,4,2)
  CALL MATADD(RADD,HSIGHT,R,2,2)
  CALL INV2X2(RADD,DDAR)
  CALL MATMUX(K,SIGHT,4,2,DDAR,2,2)
*   WRITE(11,*)'K'
*   WRITE(11,*)((K(LL,CC),CC=1,2),LL=1,4)
*** COMPUTATION OF THE STATE VECTOR "MUPLUS"
  CALL MATMUX(HMU,H,2,4,MU,4,1)
  CALL MATSUB(ZMIHMU,Z,HMU,2,1)
  CALL MATMUX(KZMIHMU,K,4,2,ZMIHMU,2,1)
  CALL MATADD(MUPLUS,MU,KZMIHMU,4,1)
*   WRITE(11,*)'MUPLUS'
*   WRITE(11,*)((MUPLUS(LL,CC),CC=1,1),LL=1,4)
*** COMPUTATION OF THE COVARIANCE MATRIX "SIGPLUS"
  CALL MATMUX(KH,K,4,2,H,2,4)
  CALL MATSUB(IDMIKH,ID,KH,4,4)
  CALL MATMUX(SIGPLUS,IDMIKH,4,4,SIG,4,4)
*   WRITE(11,*)'SIGPLUS'
*   WRITE(11,*)((SIGPLUS(LL,CC),CC=1,4),LL=1,4)
***** MOVEMENT STEP *****
*   WRITE(11,*)
*   WRITE(11,*)'MOVEMENT STEP'
*** COMPUTATION OF THE STATE VECTOR "MU"
  CALL MATMUX(MU,PHI,4,4,MUPLUS,4,1)
*   WRITE(11,*)'MU'
*   WRITE(11,*)((MU(LL,CC),CC=1,1),LL=1,4)
*** COMPUTATION OF THE COVARIANCE MATRIX "SIG"
  CALL TRANSP(PHI,4,4,PHIT)
  CALL MATMUX(PHISIG,PHI,4,4,SIGPLUS,4,4)

```

```

      CALL MATMUX(AUXSIG,PHISIG,4,4,PHIT,4,4)
      CALL MATADD(SIG,AUXSIG,Q,4,4)
*      WRITE(11,*)'SIG'
*      WRITE(11,*)((SIG(LL,CC),CC=1,4),LL=1,4)
      XE = MU(1,1)
      YE = MU(2,1)
*      WRITE(11,890)X,Y,XN,YN,XE,YE
*890   FORMAT(6(1X,F10.4))
      SCOREX = SCOREX + (X-XE)**2 /SIZE
      SCX = SCX + (X-XN)**2 /SIZE
      SCOREY = SCOREY + (Y-YE)**2 /SIZE
      SCY = SCY + (Y-YN)**2 /SIZE
200   CONTINUE
      PRINT*,SCOREX,SCX,SCOREY,SCY
      STOP
      END
*
*
      SUBROUTINE MATMUX(RESUL,MULTP,LINP,COLP,MULTR,LINR,COLR)
* SUBROUTINE FOR MATRIX MULTIPLICATION
      INTEGER LINP,COLP,LINR,COLR,K,L,M
      REAL MULTP(LINP,COLP),MULTR(LINR,COLR),RESUL(LINP,COLR),SOMA
      DO 120 K= 1,LINP
        DO 130 L= 1,COLR
          SOMA = 0.
          DO 140 M=1,COLP
            SOMA = SOMA + MULTP(K,M) * MULTR(M,L)
140      CONTINUE
          RESUL(K,L) = SOMA
130      CONTINUE
120      CONTINUE
      RETURN
      END
*
      SUBROUTINE MATADD(RES,ADD1,ADD2,NROW,NCOL)
*SUBROUTINE FOR MATRIX ADDITION
      INTEGER NROW,NCOL,K,L
      REAL RES(NROW,NCOL),ADD1(NROW,NCOL),ADD2(NROW,NCOL)
      DO 150 K=1,NROW
        DO 160 L=1,NCOL
          RES(K,L) = ADD1(K,L) + ADD2(K,L)
160      CONTINUE
150      CONTINUE
      RETURN
      END

```

```

SUBROUTINE MATSUB(RES,SUB1,SUB2,NROW,NCOL)
* SUBROUTINE FOR MATRIX SUBTRACTION
  INTEGER NROW,NCOL,K,L
  REAL RES(NROW,NCOL),SUB1(NROW,NCOL),SUB2(NROW,NCOL)
  DO 170 K=1,NROW
    DO 180 L=1,NCOL
      RES(K,L) = SUB1(K,L) - SUB2(K,L)
180  CONTINUE
170 CONTINUE
  RETURN
  END
*
SUBROUTINE TRANSP(A,AROW,ACOL,T)
* SUBROUTINE FOR MATRIX TRANSPOSITION
  INTEGER AROW,ACOL,K,L
  REAL A(AROW,ACOL),T(ACOL,AROW)
  DO 190 K=1,AROW
    DO 195 L=1,ACOL
      T(L,K) = A(K,L)
195  CONTINUE
190 CONTINUE
  RETURN
  END
*
*
SUBROUTINE INV2X2(MAT,MATINV)
* SUBROUTINE FOR 2X2 MATRIX INVERTION
  REAL MAT(2,2),MATINV(2,2),DET
  DET = (MAT(1,1) * MAT(2,2)) - (MAT(1,2) * MAT(2,1))
  IF(DET .EQ. 0.) THEN
    PRINT*, 'MATRIX IS NOT INVERTIBLE'
    STOP
  ENDIF
  MATINV(1,1) = MAT(2,2) / DET
  MATINV(1,2) = - MAT(1,2) / DET
  MATINV(2,1) = - MAT(2,1) / DET
  MATINV(2,2) = MAT(1,1) / DET
  RETURN
  END

```

APPENDIX B - FORTRAN PROGRAM FOR THE MOTION GENERATOR

PROGRAM MOTION

```
*****
* THIS PROGRAM SIMULATES 4 TARGET MOTION PATTERNS , USING THE
* NORMAL RANDOM GENERATOR OF THE PCSIMUTIL PACKAGE FROM PROF.
* MIKE BAILEY. THE RANDOM NUMBER GENERATOR IS FOR A 32 BITS PC.
* THE OUTPUT FOR FILE E:\MOTION.RES , WHERE THE 'N' IS
* SUBSTITUTED BY THE TARGET PATTERN, SHOWS 9 VARIABLES IN THE
* FOLLOWING ORDER :
* 1)TIME 2)X 3)Y 4)RANGE 5)BEARI 6)XN 7)YN 8)RANGEN 9)BEARIN
*
*****
*
***** VARIABLES *****
* ALTT  RANDOM TIME TO NEXT COURSE ALTERATION ON MODEL 4
* ALTTIME TIME OF NEXT COURSE ALTERATION ON MODEL 4 (HOURS)
* ALPHA  1/MEAN TIME BETWEEN COURSE ALTERATION ON MODEL 4
* BCOURSE TARGET BASE COURSE (DEGREES)
* BSPEED  TARGET BASE SPEED (KNOTS)
* BEARI  BEARING FROM OBSERVER TO TARGET (RADIAN)
* BEARIN  BEARING FROM OBSERVER TO TARGET WITH ERROR NOISE
*        (RADIAN)
* CHOICE  TARGET MOTION MODEL CHOICE (1 TO 4)
* COURSE  TARGET COURSE (RADIAN)
* CTIME  CORRECTED TIME TO BE USED AS AN ANGLE IN RADIAN
* DELTAT  INTERVAL TIME BETWEEN OBSERVATIONS (HOURS)
* DSEED  SEED FOR RANDOM NUMBER GENERATOR
* H      SCULPTURING PARAMETER
* I      AUXILIARY VARIABLE
* MBCD  MAXIMUM BASE COURSE DEVIATION
* MBSD  MAXIMUM BASE SPEED  DEVIATION
* MEASDEV VECTOR OF MEASURES STANDARD DEVIATION (NM,RAD)
* NOICE  NOISE ERROR CHOICE (1 OR 2)
* NORM  VECTOR WITH NORMAL RANDOM VARIABLES
* PERC  PERCENTAGE OF ACTUAL RANGE USED TO COMPUTE RANGE S.D.
* PI    AS ITSELF
* RANGE  TARGET RANGE (NAUTICAL MILES)
* RANGEN  TARGET RANGE WITH ERROR NOISE (NAUTICAL MILES)
* SPESD  STANDARD DEVIATION FOR SPEED NOISE (KNOTS)
* SPEED  TARGET SPEED (KNOTS)
* SPEEDX  TARGET SPEED IN THE X AXIS (KNOTS)
* SPEEDY  TARGET SPEED IN THE Y AXIS (KNOTS)
```

```

* SIZE  NUMBER OF POINTS TO BE GENERATED IN THE SIMULATION
* TIME  SIMULATION TIME (HOURS)
* U      AUXILIARY VARIABLE
* X      TARGET X TRUE COORDINATE (NAUTICAL MILES)
* XN     TARGET X COORDINATE WITH ERROR NOISE (NAUTICAL MILES)
* Y      TARGET Y TRUE COORDINATE (NAUTICAL MILES)
* YN     TARGET Y COORDINATE WITH ERROR NOISE (NAUTICAL MILES)
*
*****
* INITIALIZATION
  INTEGER SIZE,I,CHOICE,NOICE
  DOUBLE PRECISION DSEED
  REAL ALTT,ALTTIME,ALPHA
  REAL BCOURSE,BSPEED,BEARI,BEARIN,COURSE,CTIME,DELTAT,H
  REAL MBCD,MBSO,MEASDV(2),NORM(2),PI,PERC,SPEED,SPEEDX
  REAL SPEEDY,SPESD,RANGE,TIME,X,Y,XN,YN,U(1)
  PARAMETER (PI = 3.141593)
  DATA DELTAT/.01/ , SIZE /200/, H /0.1097466/
  TIME = 0.
  ALTTIME = 0.
  DSEED = 2332.
* USER CHOICE OF TARGET PATTERN
  WRITE(6,*)      TARGET MOTION GENERATOR'
  WRITE(6,*)
  10 WRITE(6,*)'PLEASE INPUT THE DESIRED MEASUREMENT NOISE '
  WRITE(6,*)      1 - NORMAL NOISE'
  WRITE(6,*)      2 - SCULPTURED NOISE'
  READ(6,*) NOICE
  IF((NOICE.LT.1).OR.(NOICE.GT.2)) GOTO 10
  WRITE(6,*)
  20 WRITE(6,*)'PLEASE INPUT THE DESIRED MOTION PATTERN NUMBER '
  WRITE(6,*)      1  CONSTANT COURSE AND SPEED'
  WRITE(6,*)      2  SINUSOIDAL TRANSITOR TARGET '
  WRITE(6,*)      3  SECOND ORDER GAUSS MARKOV TARGET '
  WRITE(6,*)      4  RANDOM TOUR TARGET '
  READ(6,*) CHOICE
  IF((CHOICE.LT.1).OR.(CHOICE.GT.4)) GOTO 20
  GOTO (100,200,300,400),CHOICE
*
***** GENERATION OF A MODEL 1 TARGET *****
*
  100 WRITE(6,*)'INPUT TARGET INITIAL X-Y POSITION (E.G., 0. 0.)'
  READ(6,*) X,Y
  WRITE(6,*)'INPUT TARGET BASE COURSE AND SPEED '
  READ(6,*) COURSE, SPEED
  COURSE = COURSE * PI / 180.
  SPEEDX = SPEED * COS(COURSE)

```

```

SPEEDY = SPEED * SIN(COURSE)
WRITE(6,*)'INPUT RANGE STANDARD DEVIATION, PERCENTAGE OF '
WRITE(6,*)'    ACTUAL RANGE (E.G. 5% IS 5) '
READ(6,*) PERC
PERC = PERC / 100.
OPEN(UNIT=08, FILE='D:\APL2\TEST.RES' , STATUS='NEW')
* MAIN LOOP FOR MODEL 1
DO 190 I=1,SIZE
  X = X + SPEEDX * DELTAT
  Y = Y + SPEEDY * DELTAT
  TIME = TIME + DELTAT
  RANGE = SQRT(X**2 + Y**2)
  BEARI = ATAN2(Y,X)
* GENERATION OF 2 NORMAL RANDOM BEARING AND RANGE
  CALL LNORPC(DSEED,NORM,2)
  MEASDV(1) = PERC * RANGE
  MEASDV(2) = .01
  IF(NOICE.EQ.1) THEN
    RANGEN = RANGE + NORM(1) * MEASDV(1)
    BEARIN = BEARI + NORM(2) * MEASDV(2)
  ELSE
    RANGEN = RANGE + ((NORM(1) * (1. + H * NORM(1) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(1)
    BEARIN = BEARI + ((NORM(2) * (1. + H * NORM(2) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(2)
  ENDIF
  XN = RANGEN * COS(BEARIN)
  YN = RANGEN * SIN(BEARIN)
  WRITE(08,115) TIME,X,Y,RANGE,180.*BEARI/PI,XN,YN,RANGEN,
C      180.*BEARIN/PI
115  FORMAT(1X,F5.2,3(1X,F6.2),1X,F11.6,2(1X,F8.4),
C      1X,F10.6,1X,F11.6)
190 CONTINUE
  GOTO 900
*
***** GENERATION OF A MODEL 2 TARGET *****
*
200 WRITE(6,*)'INPUT TARGET INITIAL X-Y POSITION (E.G., 0. 0.)'
  READ(6,*) X,Y
  WRITE(6,*)'INPUT TARGET BASE COURSE AND SPEED '
  READ(6,*) BCOURSE,BSPEED
  BCOURSE = BCOURSE * PI / 180.
  WRITE(6,*)'INPUT MAXIMUM DEVIATION FROM BASE COURSE (E.G., +/-
C 45 IS 45)'
  READ(6,*) MBCD
  MBCD = MBCD * PI / 180.
  WRITE(6,*)'INPUT MAXIMUM DEVIATION FROM BASE SPEED (E.G., +/- 2

```

```

C KTS IS 2 )
READ(6,*) MBSD
WRITE(6,*)'INPUT RANGE STANDARD DEVIATION, PERCENTAGE OF '
WRITE(6,*)' ACTUAL RANGE (E.G. 5% IS 5) '
READ(6,*) PERC
PERC = PERC / 100.
OPEN(UNIT=09, FILE='D:\APL2\MOTIO2.RES' , STATUS='NEW')
DO 290 I = 1, SIZE
  CTIME = AMOD(TIME,2*PI)
  COURSE = BCOURSE + MBSD * SIN(SPEED*CTIME)
  SPEED = BSPEED + MBSD * SIN(CTIME)
  X = X + SPEED * COS(COURSE) * DELTAT
  Y = Y + SPEED * SIN(COURSE) * DELTAT
  RANGE = SQRT(X**2 + Y**2)
  BEARI = ATAN2(Y,X)
* GENERATION OF 2 NORMAL RANDOM BEARING AND RANGE
  CALL LNORPC(DSEED,NORM,2)
  MEASDV(1) = PERC * RANGE
  MEASDV(2) = .01
  IF(VOICE.EQ.1) THEN
    RANGEN = RANGE + NORM(1) * MEASDV(1)
    BEARIN = BEARI + NORM(2) * MEASDV(2)
  ELSE
    RANGEN = RANGE + ((NORM(1) * (1. + H * NORM(1) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(1)
    BEARIN = BEARI + ((NORM(2) * (1. + H * NORM(2) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(2)
  ENDIF
  XN = RANGEN * COS(BEARIN)
  YN = RANGEN * SIN(BEARIN)
  WRITE(09,115) TIME,X,Y,RANGE,180.*BEARI/PI,XN,YN,RANGEN,
C      180.*BEARIN/PI
  TIME = TIME + DELTAT
290 CONTINUE
  GOTO 900
*
***** GENERATION OF A MODEL 3 TARGET *****
*
300 WRITE(6,*)'INPUT TARGET INITIAL X-Y POSITION (E.G., 0. 0.)'
  READ(6,*) X,Y
  WRITE(6,*)'INPUT TARGET INITIAL COURSE (DEGREES) AND SPEED (KTS)'
  READ(6,*) BCOURSE,BSPEED
  BCOURSE = BCOURSE * PI / 180.
  WRITE(6,*)'INPUT SPEED STANDARD DEVIATION FOR NORMAL NOISE '
  READ(6,*) SPESD
  WRITE(6,*)'INPUT RANGE STANDARD DEVIATION, PERCENTAGE OF '
  WRITE(6,*)' ACTUAL RANGE (E.G. 5% IS 5) '

```



```

READ(6,*) PERC
PERC = PERC / 100.
SPEEDX = BSPEED * COS(BCOURSE)
SPEEDY = BSPEED * SIN(BCOURSE)
OPEN(UNIT=10, FILE='D:\APL2\MOTTO3.RES' , STATUS='NEW')
DO 390 I = 1,SIZE
  CALL LNORPC(DSEED,NORM,2)
  SPEEDX = .8 * SPEEDX + NORM(1) * SPESD
  SPEEDY = .8 * SPEEDY + NORM(2) * SPESD
  X = X + SPEEDX * DELTAT
  Y = Y + SPEEDY * DELTAT
  RANGE = SQRT(X**2 + Y**2)
  BEARI = ATAN2(Y,X)
* GENERATION OF 2 NORMAL RANDOM BEARING AND RANGE
  CALL LNORPC(DSEED,NORM,2)
  MEASDV(1) = PERC * RANGE
  MEASDV(2) = .01
  IF(VOICE.EQ.1) THEN
    RANGEN = RANGE + NORM(1) * MEASDV(1)
    BEARIN = BEARI + NORM(2) * MEASDV(2)
  ELSE
    RANGEN = RANGE + ((NORM(1) * (1. + H * NORM(1) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(1)
    BEARIN = BEARI + ((NORM(2) * (1. + H * NORM(2) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(2)
  ENDIF
  XN = RANGEN * COS(BEARIN)
  YN = RANGEN * SIN(BEARIN)
  WRITE(10,115) TIME,X,Y,RANGE,180.*BEARI/PI,XN,YN,RANGEN,
C      180.*BEARIN/PI
  TIME = TIME + DELTAT
390 CONTINUE
  GOTO 900
*
***** GENERATION OF A MODEL 4 TARGET *****
*
400 WRITE(6,*)'INPUT TARGET INITIAL X-Y POSITION (E.G., 0. 0.)'
  READ(6,*) X,Y
  WRITE(6,*)'INPUT MEAN TIME (HRS) BETWEEN COURSE CHANGES '
  READ(6,*) ALPHA
  ALPHA = 1. / ALPHA
  WRITE(6,*)'INPUT TARGET SPEED (KTS)'
  READ(6,*) SPEED
  WRITE(6,*)'INPUT RANGE STANDARD DEVIATION AS A PERCENTAGE OF'
  WRITE(6,*)'    ACTUAL RANGE (E.G. 5% IS 5)
  READ(6,*) PERC
  PERC = PERC / 100.

```

```

      OPEN(UNIT=08, FILE='D:\APL2\MOTIO4.RES' , STATUS='NEW')
* MAIN LOOP FOR MODEL 4
      DO 490 I=1,SIZE
        IF(ALTTIME .LE. TIME) THEN
          CALL LRNDPC(DSEED,U,1)
          COURSE = 2 * PI * U(1)
          CALL EXPOPC(DSEED,ALTT,ALPHA)
          ALTTIME = ALTTIME + ALTT
          SPEEDX = SPEED * COS(COURSE)
          SPEEDY = SPEED * SIN(COURSE)
        ENDIF
        X = X + SPEEDX * DELTAT
        Y = Y + SPEEDY * DELTAT
        TIME = TIME + DELTAT
        RANGE = SQRT(X**2 + Y**2)
        BEARI = ATAN2(Y,X)
* GENERATION OF 2 NORMAL RANDOM BEARING AND RANGE
        CALL LNORPC(DSEED,NORM,2)
        MEASDV(1) = PERC * RANGE
        MEASDV(2) = .01
        IF(VOICE.EQ.1) THEN
          RANGEN = RANGE + NORM(1) * MEASDV(1)
          BEARIN = BEARI + NORM(2) * MEASDV(2)
        ELSE
          RANGEN = RANGE + ((NORM(1) * (1. + H * NORM(1) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(1)
          BEARIN = BEARI + ((NORM(2) * (1. + H * NORM(2) ** 2))
C      / ((1. + 6*H + 15* H**2)**.5)) * MEASDV(2)
        ENDIF
        XN = RANGEN * COS(BEARIN)
        YN = RANGEN * SIN(BEARIN)
        WRITE(08,115) TIME,X,Y,RANGE,180.*BEARI/PI,XN,YN,RANGEN,
C      180.*BEARIN/PI
      490 CONTINUE
      900 STOP
      END
*
* THIS SUBROUTINE WILL GENERATE A VECTOR OF NORMAL RANDOM
* VARIABLES ACCORDING TO THE SINE-COSINE METHOD
*
      SUBROUTINE LNORPC(DSEED,A,N)
      INTEGER N,I,IND
      REAL A(N),U(2),S,W,U1,XSTAR
      DOUBLE PRECISION DSEED,PI
      DATA PI/3.14159265358979D0/
      IND = 1
      DO 100 I=1,N

```

```

      IND = -IND
      IF (IND.GE.0) GOTO 20
10   CALL LRNDPC(DSEED,U,2)
      S = SQRT(-2*ALOG(U(1)))
      W = 2*PI*U(2)
      XSTAR = S*COS(W)
      A(I) = S*SIN(W)
      GOTO 100
20   A(I) = XSTAR
100 CONTINUE
      RETURN
      END
*
*
      SUBROUTINE LRNDPC (DSEED,U,N)
      INTEGER      N, I
      REAL         U(N)
      DOUBLE PRECISION  D31M1, DSEED, D31
C     D31M1=2**31 - 1
C     D31  =2**31
      DATA D31M1/2147483647.D0/
      DATA D31  /2147483648.D0/
      DO 5 I=1,N
C     DSEED = DMOD(950706376.D0*DSEED,D31M1)
      DSEED = DMOD(16807.D0*DSEED,D31M1)
5    U(I) = DSEED / D31
      RETURN
      END
*
* THIS SUBROUTINE GENERATES 1 EXPONENTIAL(LAMBDA) RANDOM
* NUMBER
      SUBROUTINE EXPOPC(DSEED,EXPO,LAMBDA)
      DOUBLE PRECISION DSEED
      REAL EXPO,LAMBDA,UNI(1)
      CALL LRNDPC(DSEED,UNI,1)
      A = 1.0 - UNI(1)
      EXPO = - ALOG(A) / LAMBDA
      RETURN
      END
*

```

LIST OF REFERENCES

1. Vilhena, R., *Automacao de Sistemas Navais*, Capitulo 3, Apostila do Curso da Escola Naval.
2. Naval Electronics Systems Command, *Over-the-horizon / Detection, Classification and Targeting. System Level Specification. Ship tracking algorithm*. Section 6 of notes excerpted by Barker, W.H., Undated.
3. Belkin, B., *The Ornstein-Uhlenbeck Displacement Process as a Model for Target Motion*, DHWA Memorandum to Applied Physics Laboratory, Johns Hopkins University, 1 February 1978.
4. Wagner, D.H., *Naval Tactical Decision Aids*, Technical Report NPS55-89-011, Naval Postgraduate School, September 1989.
5. Bar-Shalom, Y., and Fortmann, T.E., *Tracking and Data Association*, Academic Press, 1988.
6. Washburn, A.R., *A Short Introduction to Kalman Filters*, Naval Postgraduate School.
7. Farina, A., Studer, F.A., *Radar Data Processing*, Vol. 1, Research Studies Press., 1985.
8. Waltz, E., *Multisensor Data Fusion*, Artech House Inc., 1990.
9. Bailey, M. P., *Simutil Fortran*, Simulation Utility Subroutines Unpublished Lecture Notes, System Simulation, Department of Operations Research, Naval Postgraduate School, Monterey, CA 93943-5000. 1990.
10. Louchard, G., Latouche, G., *Probability Theory and Computer Science*, Chapter 3, Academic Press, 1983.
11. Box, G.E., Hunter, J.S., and Hunter, W.L., *Statistics for Experimenters*, Wiley Interscience, 1978.

INITIAL DISTRIBUTION LIST

- | | |
|--|---|
| 1. Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22304-6105 | 2 |
| 2. Library, code 52
Naval Postgraduate School
Monterey, California 93943-5002 | 2 |
| 3. Professor Donald P. Gaver, Code OR/GV
Naval Postgraduate School
Monterey, California 93943-5000 | 2 |
| 4. Jose B. Souza Neto
Centro de Analise de Sistemas Navais
Cidade Militar 480, Rio de Janeiro, Br | 3 |